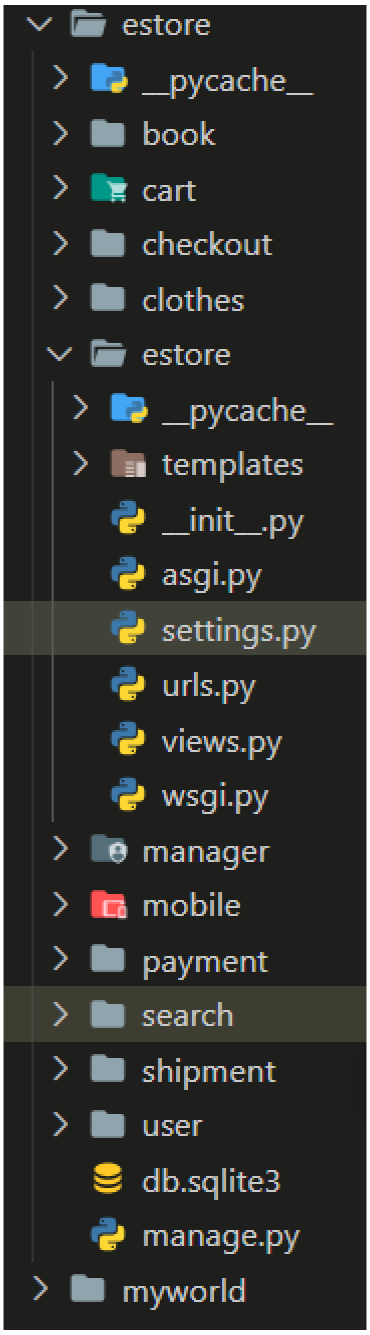
Họ và tên: Nguyễn Hữu Thịnh – B20DCCN672

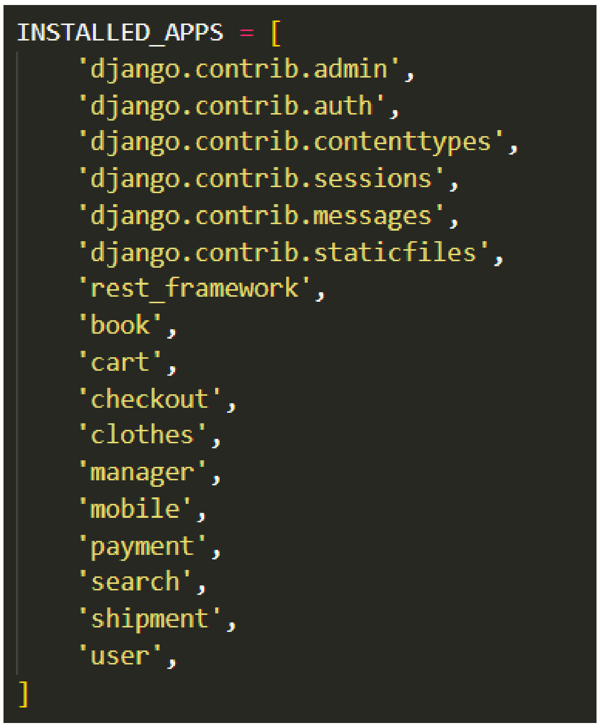
Bài tập 4 – S.A&D

Create 10 applications/services in Django: user, manager, cart, check out & order, search, book, mobile, clothes, shipment, payment



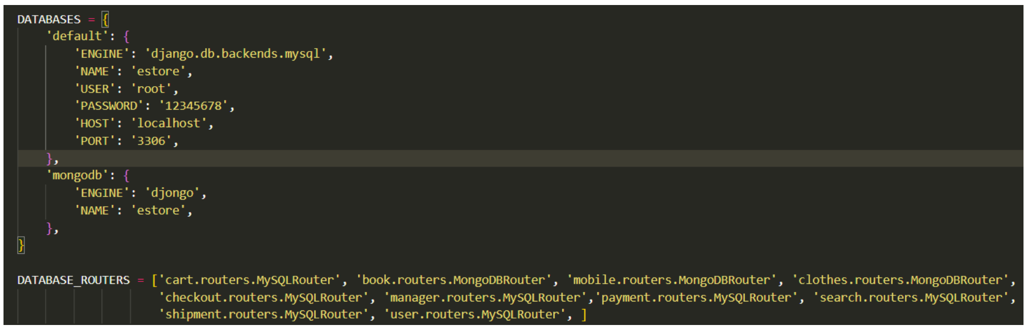
```
estore
  __pycache__
  book
  cart
  checkout
  clothes
  estore
    __pycache__
    templates
    __init__.py
    asgi.py
    settings.py
    urls.py
    views.py
    wsgi.py
  manager
  mobile
  payment
  search
  shipment
  user
  db.sqlite3
  manage.py
myworld
```

2. Khai báo app



```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'book',
    'cart',
    'checkout',
    'clothes',
    'manager',
    'mobile',
    'payment',
    'search',
    'shipment',
    'user',
]
```

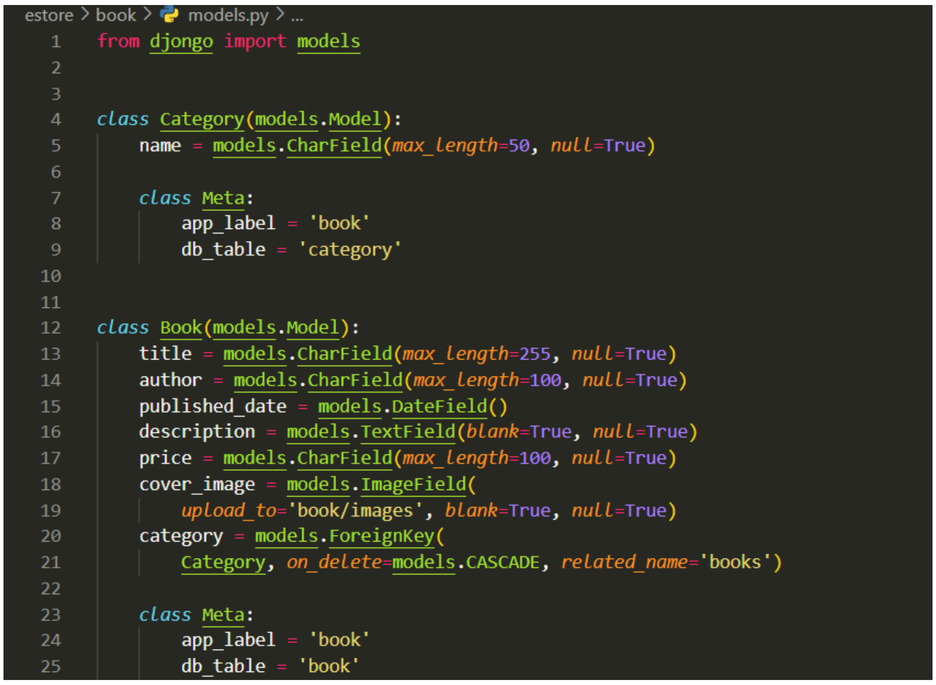
3. Config Database và Routers



```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'estore',
        'USER': 'root',
        'PASSWORD': '12345678',
        'HOST': 'localhost',
        'PORT': '3306',
    },
    'mongodb': {
        'ENGINE': 'djongo',
        'NAME': 'estore',
    },
}

DATABASE_ROUTERS = ['cart.routers.MySQLRouter', 'book.routers.MongoDBRouter', 'mobile.routers.MongoDBRouter', 'clothes.routers.MongoDBRouter',
                    'checkout.routers.MySQLRouter', 'manager.routers.MySQLRouter','payment.routers.MySQLRouter', 'search.routers.MySQLRouter',
                    'shipment.routers.MySQLRouter', 'user.routers.MySQLRouter', ]
```

4. App book

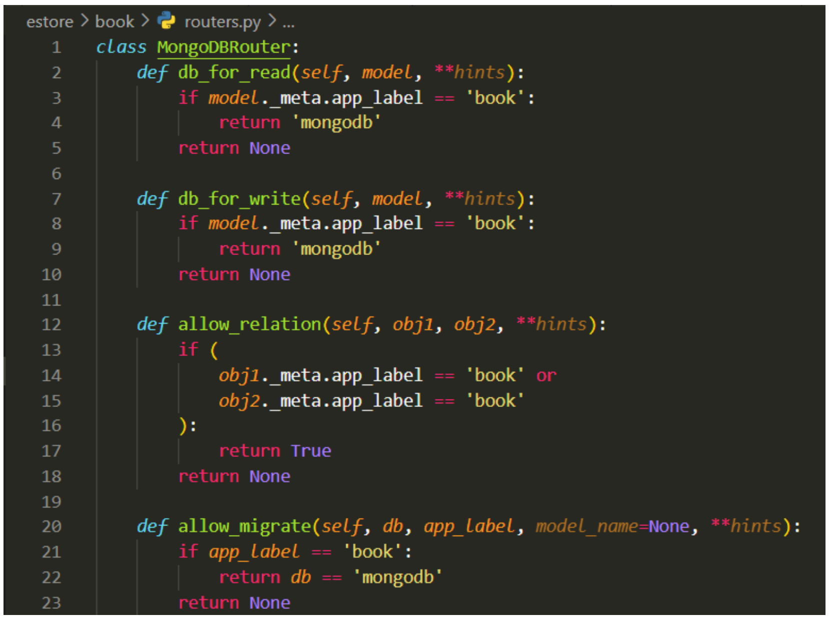
models.py



estore > book > models.py > ...

```python
from djongo import models


class Category(models.Model):
    name = models.CharField(max_length=50, null=True)

    class Meta:
        app_label = 'book'
        db_table = 'category'


class Book(models.Model):
    title = models.CharField(max_length=255, null=True)
    author = models.CharField(max_length=100, null=True)
    published_date = models.DateField()
    description = models.TextField(blank=True, null=True)
    price = models.CharField(max_length=100, null=True)
    cover_image = models.ImageField(
        upload_to='book/images', blank=True, null=True)
    category = models.ForeignKey(
        Category, on_delete=models.CASCADE, related_name='books')

    class Meta:
        app_label = 'book'
        db_table = 'book'
```

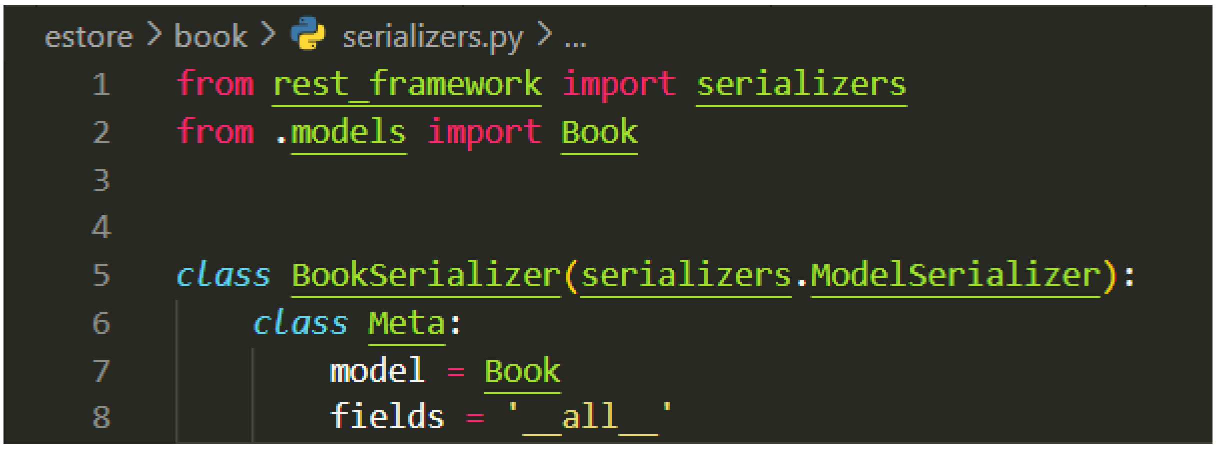
Routers.py



estore > book > routers.py > ...

```python
class MongoDBRouter:
    def db_for_read(self, model, **hints):
        if model._meta.app_label == 'book':
            return 'mongodb'
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label == 'book':
            return 'mongodb'
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label == 'book' or
            obj2._meta.app_label == 'book'
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == 'book':
            return db == 'mongodb'
        return None
```

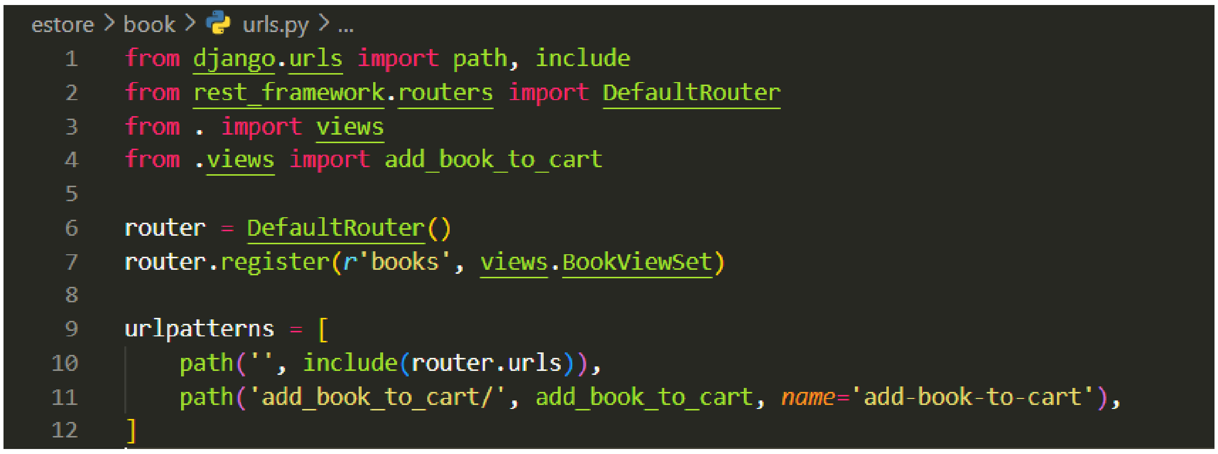
Serializers.py



estore > book > serializers.py > ...

```python
from rest_framework import serializers
from .models import Book


class BookSerializer(serializers.ModelSerializer):
    class Meta:
        model = Book
        fields = '__all__'
```

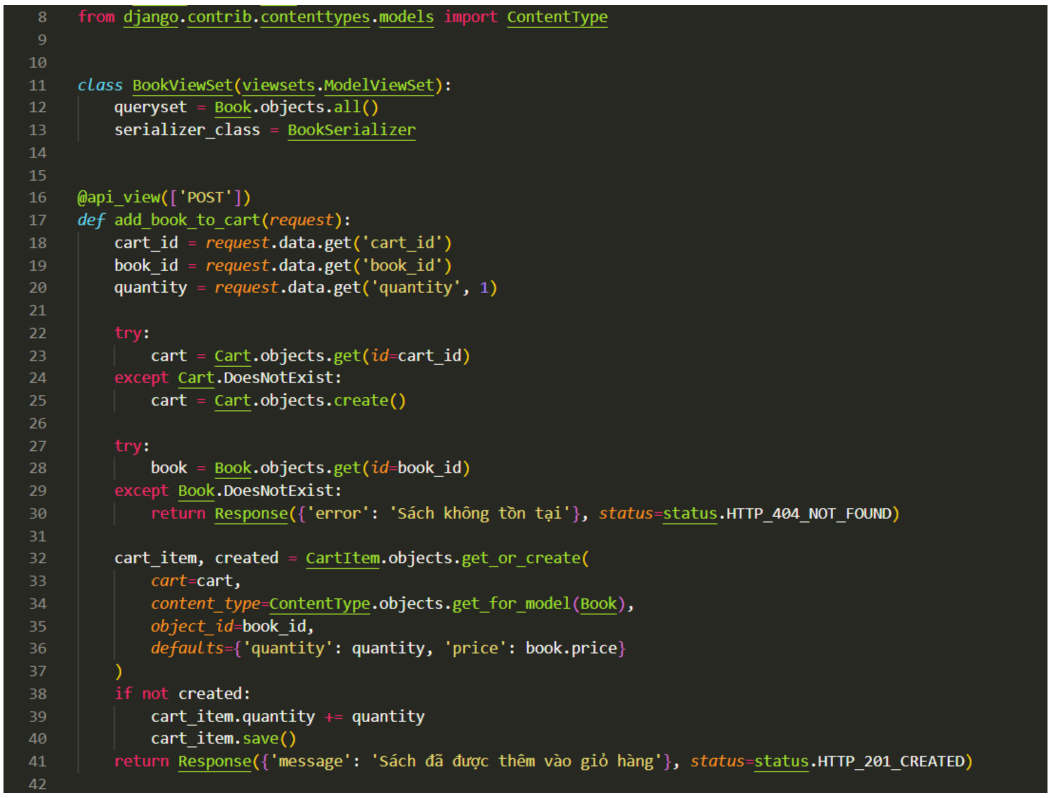
Urls.py



estore > book > urls.py > ...

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from . import views
from .views import add_book_to_cart

router = DefaultRouter()
router.register(r'books', views.BookViewSet)

urlpatterns = [
    path('', include(router.urls)),
    path('add_book_to_cart/', add_book_to_cart, name='add-book-to-cart'),
]
```

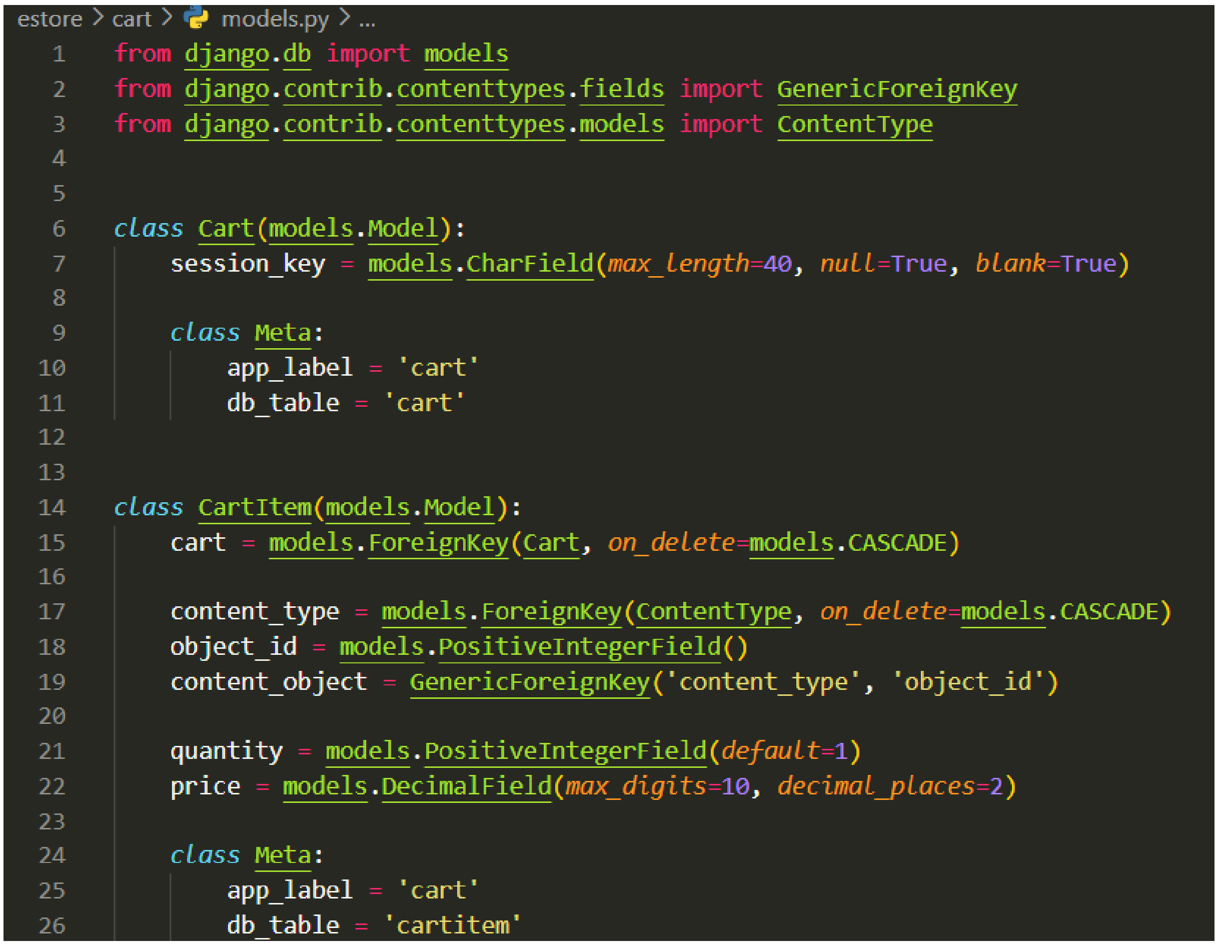
Views.py



```python
from django.contrib.contenttypes.models import ContentType


class BookViewSet(viewsets.ModelViewSet):
    queryset = Book.objects.all()
    serializer_class = BookSerializer


@api_view(['POST'])
def add_book_to_cart(request):
    cart_id = request.data.get('cart_id')
    book_id = request.data.get('book_id')
    quantity = request.data.get('quantity', 1)

    try:
        cart = Cart.objects.get(id=cart_id)
    except Cart.DoesNotExist:
        cart = Cart.objects.create()

    try:
        book = Book.objects.get(id=book_id)
    except Book.DoesNotExist:
        return Response({'error': 'Sách không tồn tại'}, status=status.HTTP_404_NOT_FOUND)

    cart_item, created = CartItem.objects.get_or_create(
        cart=cart,
        content_type=ContentType.objects.get_for_model(Book),
        object_id=book_id,
        defaults={'quantity': quantity, 'price': book.price}
    )
    if not created:
        cart_item.quantity += quantity
        cart_item.save()
    return Response({'message': 'Sách đã được thêm vào giỏ hàng'}, status=status.HTTP_201_CREATED)

```

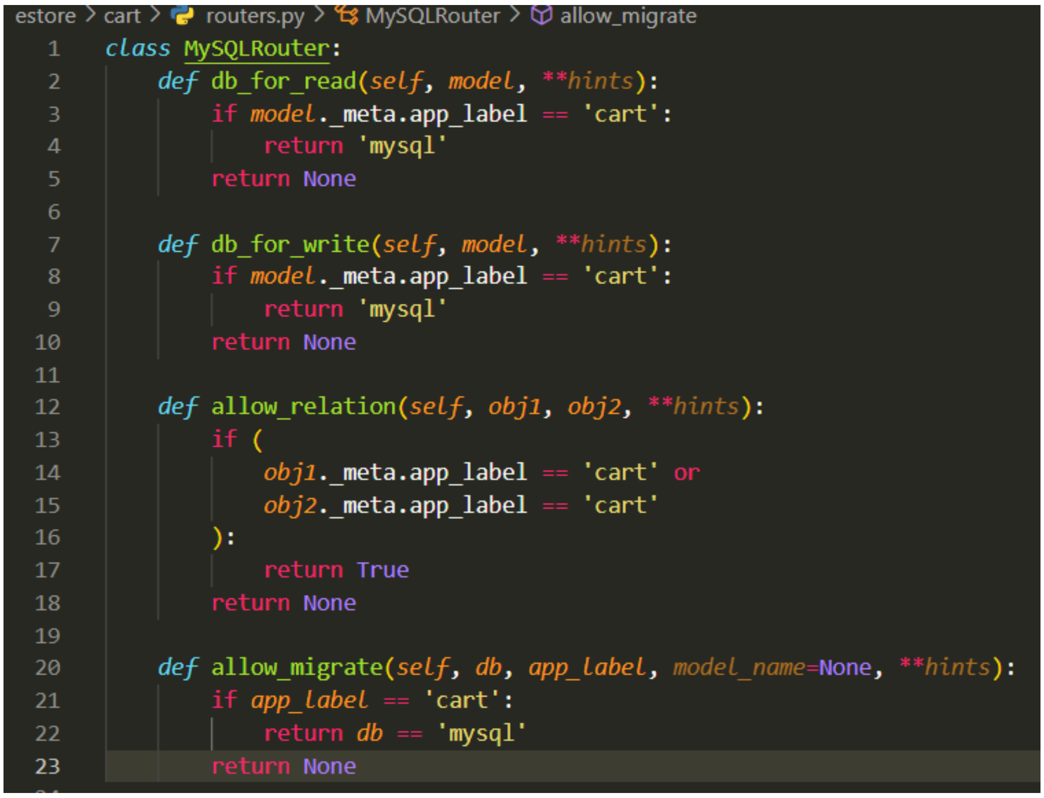
5. App cart

Models.py



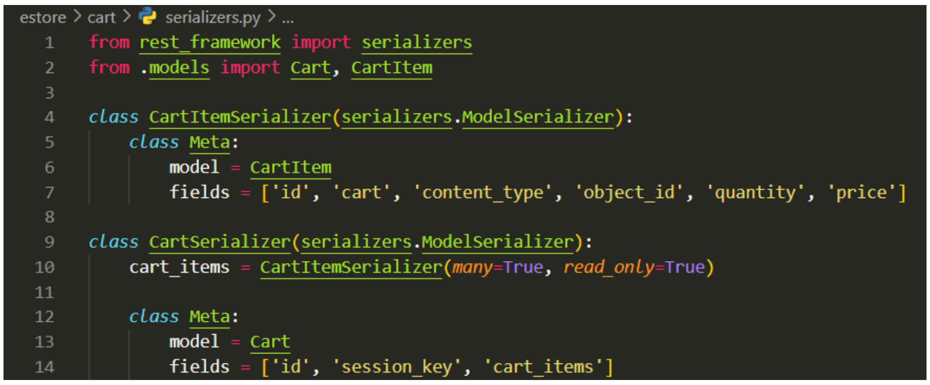
```python
from django.db import models
from django.contrib.contenttypes.fields import GenericForeignKey
from django.contrib.contenttypes.models import ContentType


class Cart(models.Model):
    session_key = models.CharField(max_length=40, null=True, blank=True)

    class Meta:
        app_label = 'cart'
        db_table = 'cart'


class CartItem(models.Model):
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)

    content_type = models.ForeignKey(ContentType, on_delete=models.CASCADE)
    object_id = models.PositiveIntegerField()
    content_object = GenericForeignKey('content_type', 'object_id')

    quantity = models.PositiveIntegerField(default=1)
    price = models.DecimalField(max_digits=10, decimal_places=2)

    class Meta:
        app_label = 'cart'
        db_table = 'cartitem'
```

Routers.py


estore > cart > routers.py > MySQLRouter > allow_migrate

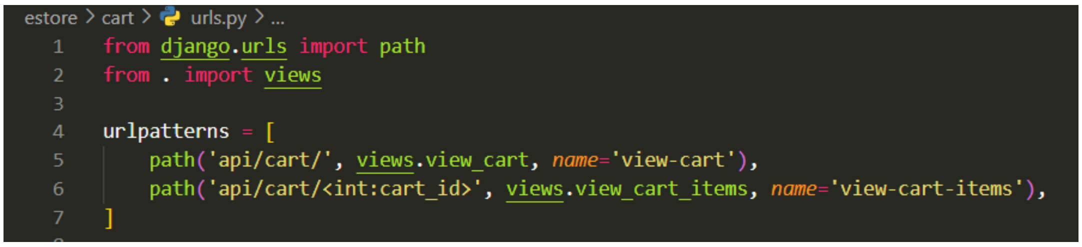
```python
class MySQLRouter:
    def db_for_read(self, model, **hints):
        if model._meta.app_label == 'cart':
            return 'mysql'
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label == 'cart':
            return 'mysql'
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label == 'cart' or
            obj2._meta.app_label == 'cart'
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == 'cart':
            return db == 'mysql'
        return None
```

Serializers.py


estore > cart > serializers.py > ...

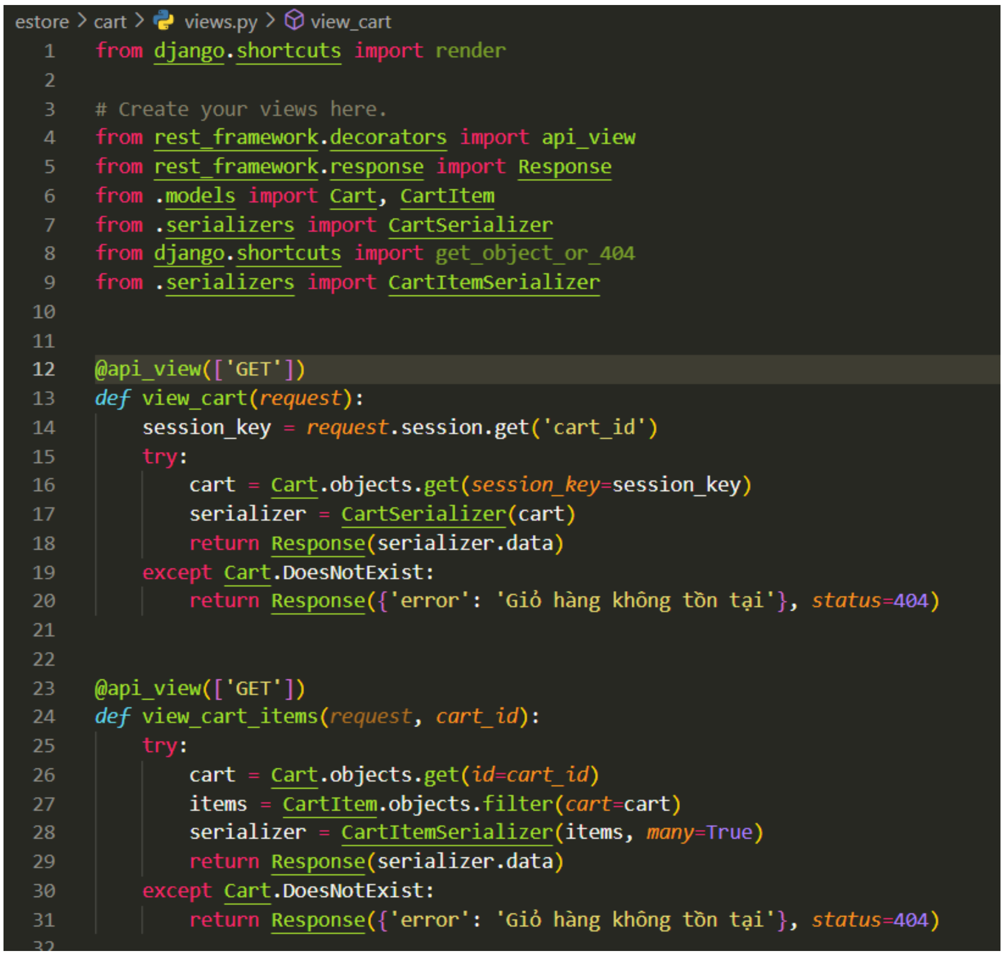
```python
from rest_framework import serializers
from .models import Cart, CartItem

class CartItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = CartItem
        fields = ['id', 'cart', 'content_type', 'object_id', 'quantity', 'price']

class CartSerializer(serializers.ModelSerializer):
    cart_items = CartItemSerializer(many=True, read_only=True)

    class Meta:
        model = Cart
        fields = ['id', 'session_key', 'cart_items']
```

Urls.py


estore > cart > urls.py > ...

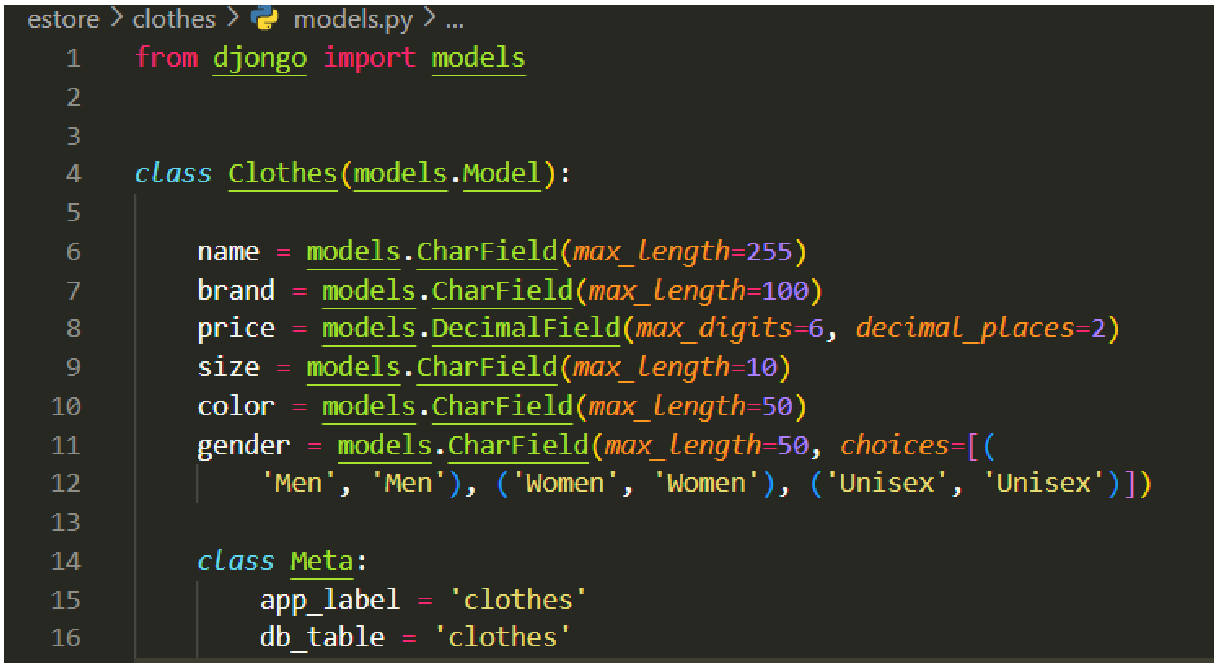
```python
from django.urls import path
from . import views

urlpatterns = [
    path('api/cart/', views.view_cart, name='view-cart'),
    path('api/cart/<int:cart_id>', views.view_cart_items, name='view-cart-items'),
]
```

Views.py

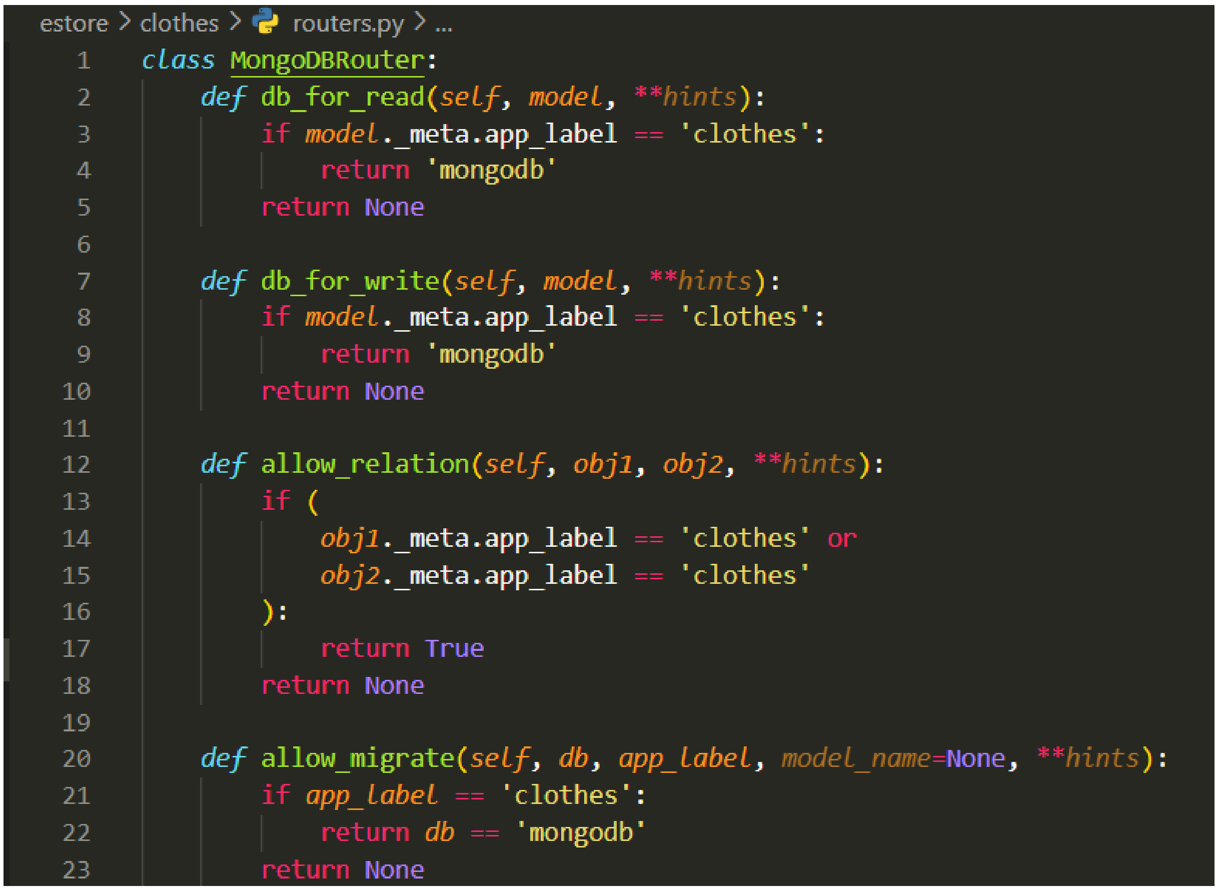


```python
from django.shortcuts import render

# Create your views here.
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .models import Cart, CartItem
from .serializers import CartSerializer
from django.shortcuts import get_object_or_404
from .serializers import CartItemSerializer


@api_view(['GET'])
def view_cart(request):
    session_key = request.session.get('cart_id')
    try:
        cart = Cart.objects.get(session_key=session_key)
        serializer = CartSerializer(cart)
        return Response(serializer.data)
    except Cart.DoesNotExist:
        return Response({'error': 'Giỏ hàng không tồn tại'}, status=404)


@api_view(['GET'])
def view_cart_items(request, cart_id):
    try:
        cart = Cart.objects.get(id=cart_id)
        items = CartItem.objects.filter(cart=cart)
        serializer = CartItemSerializer(items, many=True)
        return Response(serializer.data)
    except Cart.DoesNotExist:
        return Response({'error': 'Giỏ hàng không tồn tại'}, status=404)
```

6. App clothes

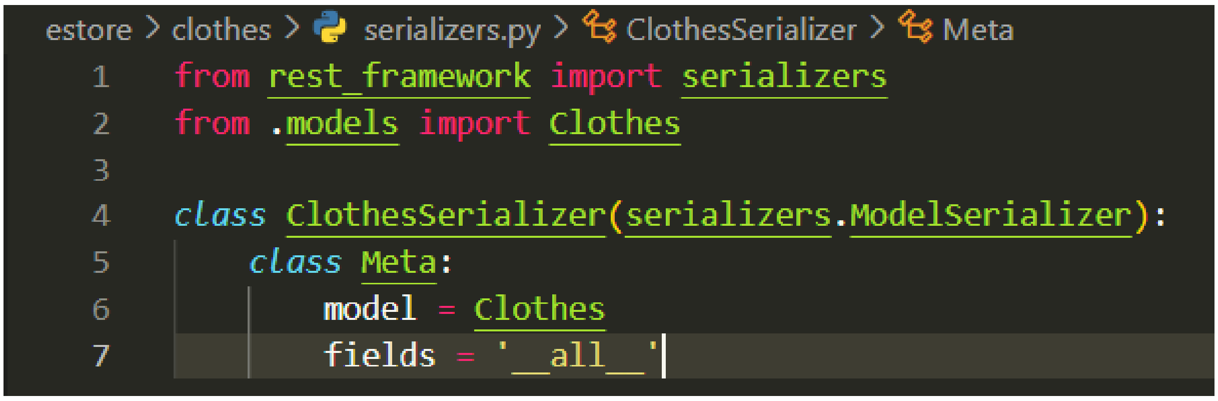


```python
from djongo import models


class Clothes(models.Model):

    name = models.CharField(max_length=255)
    brand = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=6, decimal_places=2)
    size = models.CharField(max_length=10)
    color = models.CharField(max_length=50)
    gender = models.CharField(max_length=50, choices=[(
        'Men', 'Men'), ('Women', 'Women'), ('Unisex', 'Unisex')])

    class Meta:
        app_label = 'clothes'
        db_table = 'clothes'
```



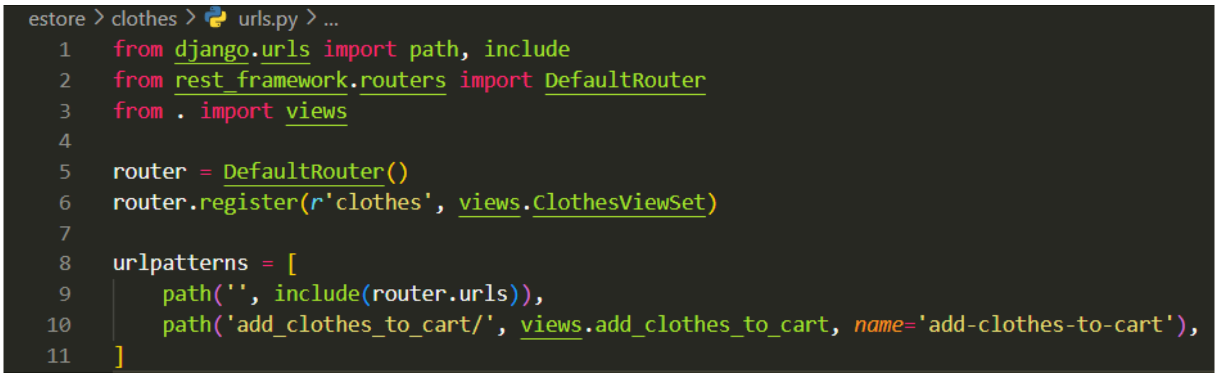
estore > clothes > routers.py > ...

```python
class MongoDBRouter:
    def db_for_read(self, model, **hints):
        if model._meta.app_label == 'clothes':
            return 'mongodb'
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label == 'clothes':
            return 'mongodb'
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label == 'clothes' or
            obj2._meta.app_label == 'clothes'
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == 'clothes':
            return db == 'mongodb'
        return None
```



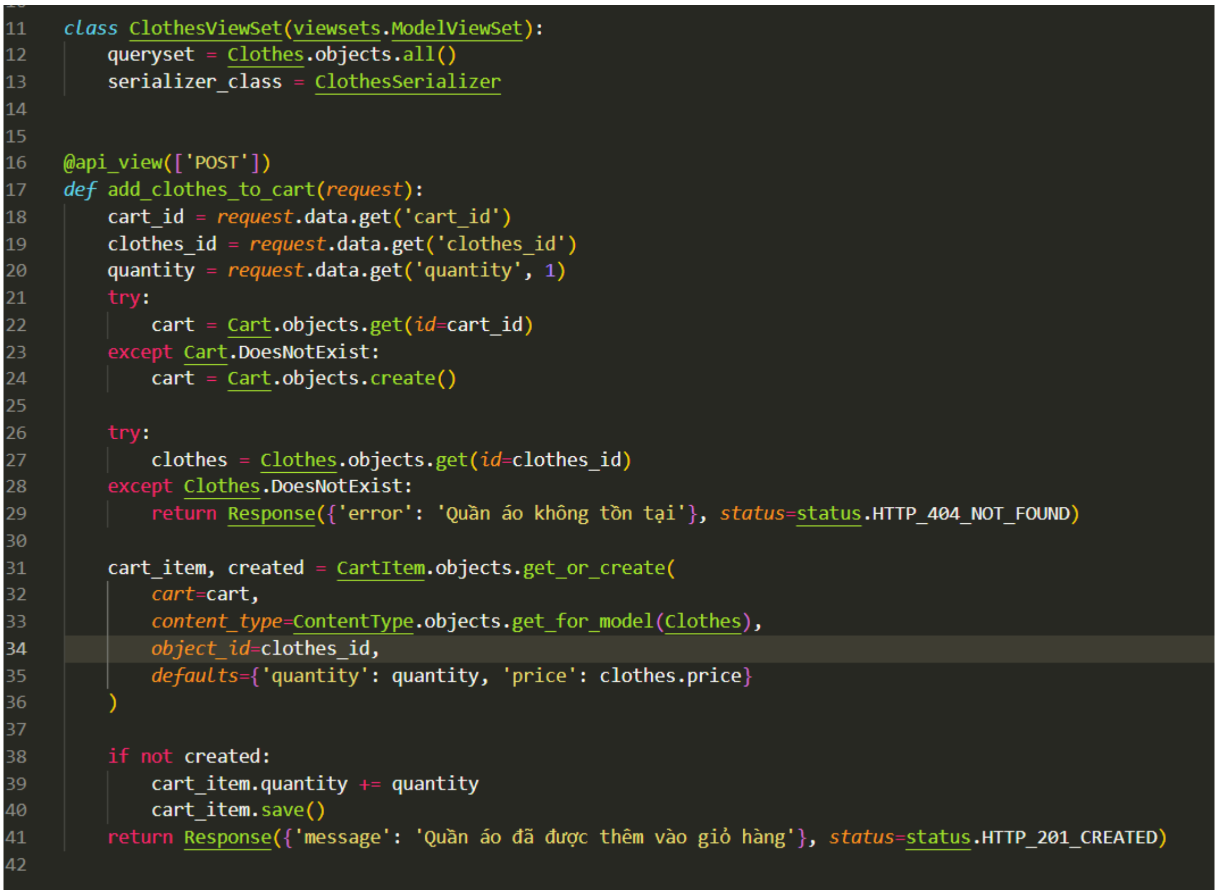
estore > clothes > serializers.py > ClothesSerializer > Meta

```python
from rest_framework import serializers
from .models import Clothes

class ClothesSerializer(serializers.ModelSerializer):
    class Meta:
        model = Clothes
        fields = '__all__'
```



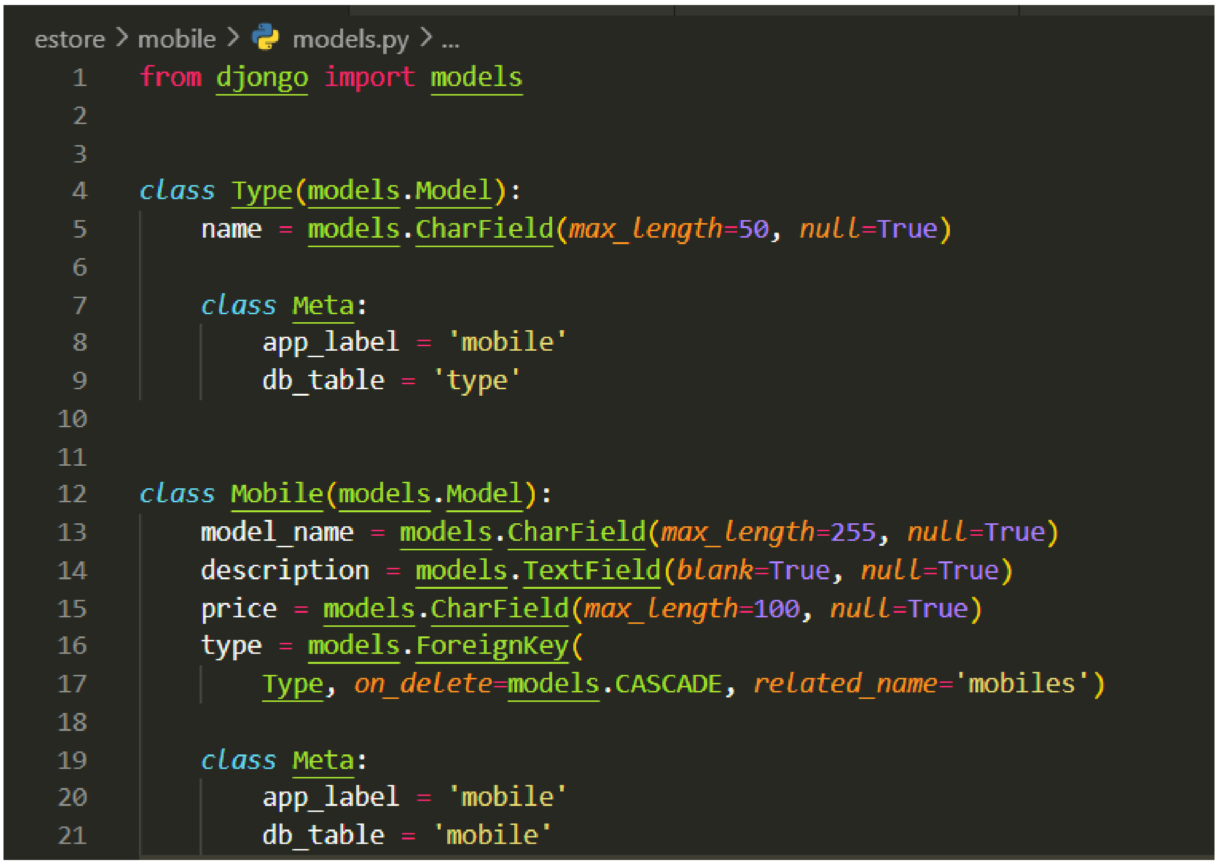
estore > clothes > urls.py > ...

```python
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from . import views

router = DefaultRouter()
router.register(r'clothes', views.ClothesViewSet)

urlpatterns = [
    path('', include(router.urls)),
    path('add_clothes_to_cart/', views.add_clothes_to_cart, name='add-clothes-to-cart'),
]
```



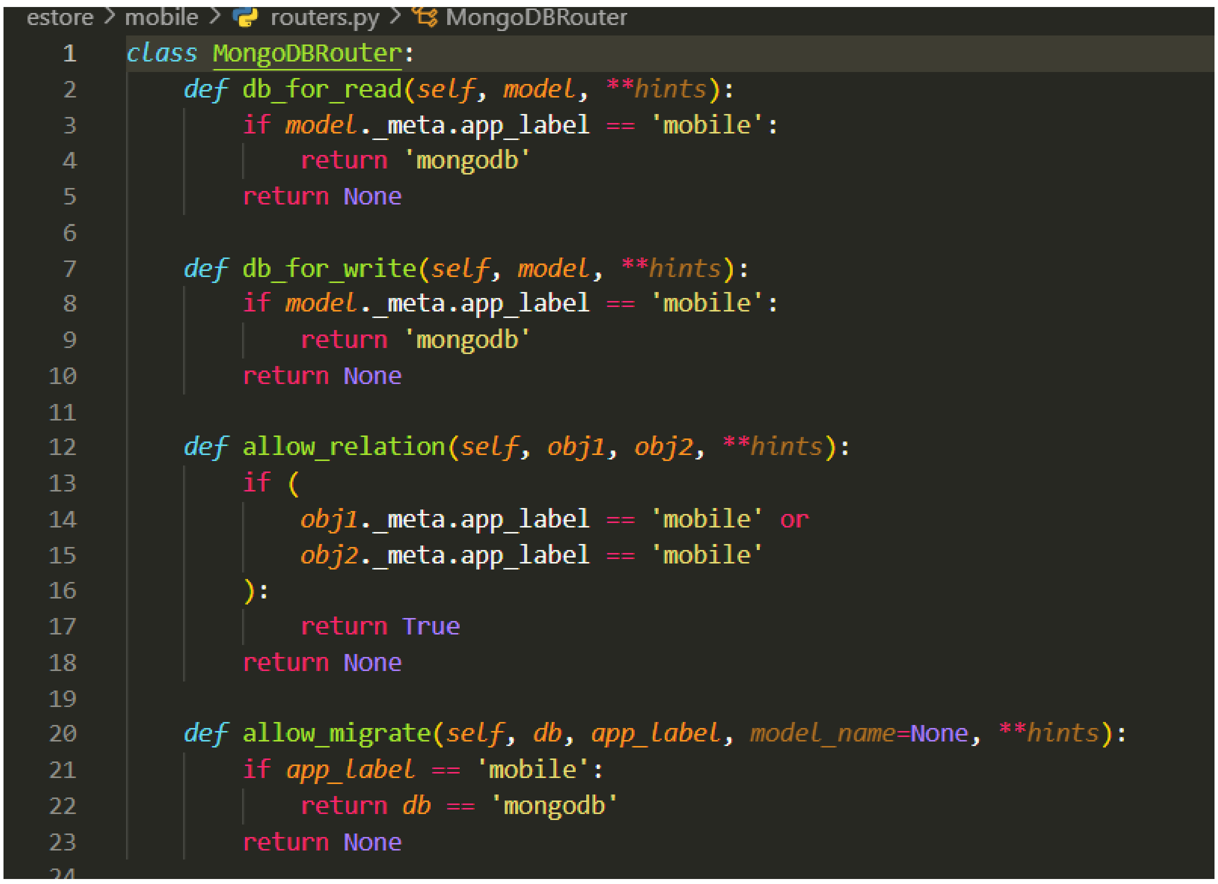
```
class ClothesViewSet(viewsets.ModelViewSet):
    queryset = Clothes.objects.all()
    serializer_class = ClothesSerializer


@api_view(['POST'])
def add_clothes_to_cart(request):
    cart_id = request.data.get('cart_id')
    clothes_id = request.data.get('clothes_id')
    quantity = request.data.get('quantity', 1)
    try:
        cart = Cart.objects.get(id=cart_id)
    except Cart.DoesNotExist:
        cart = Cart.objects.create()

    try:
        clothes = Clothes.objects.get(id=clothes_id)
    except Clothes.DoesNotExist:
        return Response({'error': 'Quần áo không tồn tại'}, status=status.HTTP_404_NOT_FOUND)

    cart_item, created = CartItem.objects.get_or_create(
        cart=cart,
        content_type=ContentType.objects.get_for_model(Clothes),
        object_id=clothes_id,
        defaults={'quantity': quantity, 'price': clothes.price}
    )

    if not created:
        cart_item.quantity += quantity
        cart_item.save()
    return Response({'message': 'Quần áo đã được thêm vào giỏ hàng'}, status=status.HTTP_201_CREATED)
```

7. App mobile



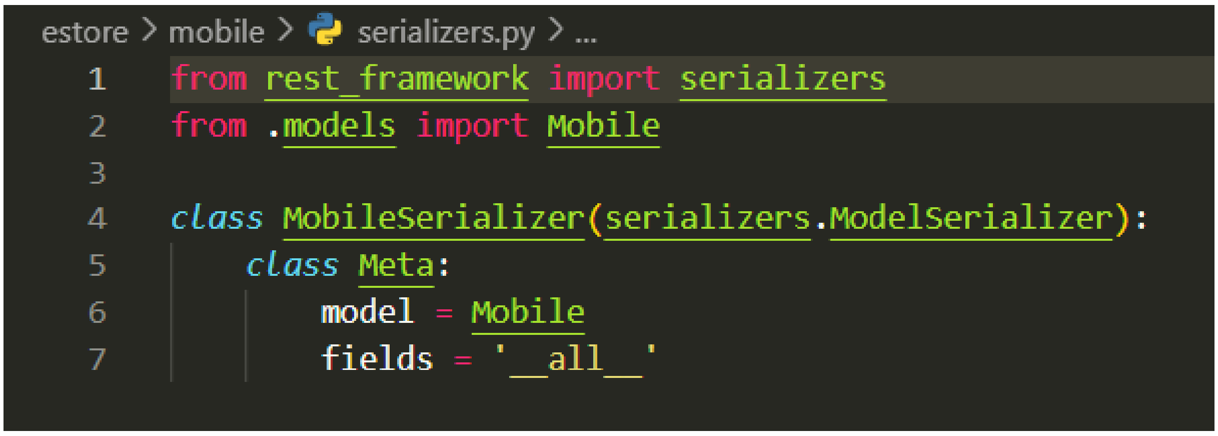
estore > mobile > models.py > ...

```
from djongo import models


class Type(models.Model):
    name = models.CharField(max_length=50, null=True)

    class Meta:
        app_label = 'mobile'
        db_table = 'type'


class Mobile(models.Model):
    model_name = models.CharField(max_length=255, null=True)
    description = models.TextField(blank=True, null=True)
    price = models.CharField(max_length=100, null=True)
    type = models.ForeignKey(
        Type, on_delete=models.CASCADE, related_name='mobiles')

    class Meta:
        app_label = 'mobile'
        db_table = 'mobile'
```



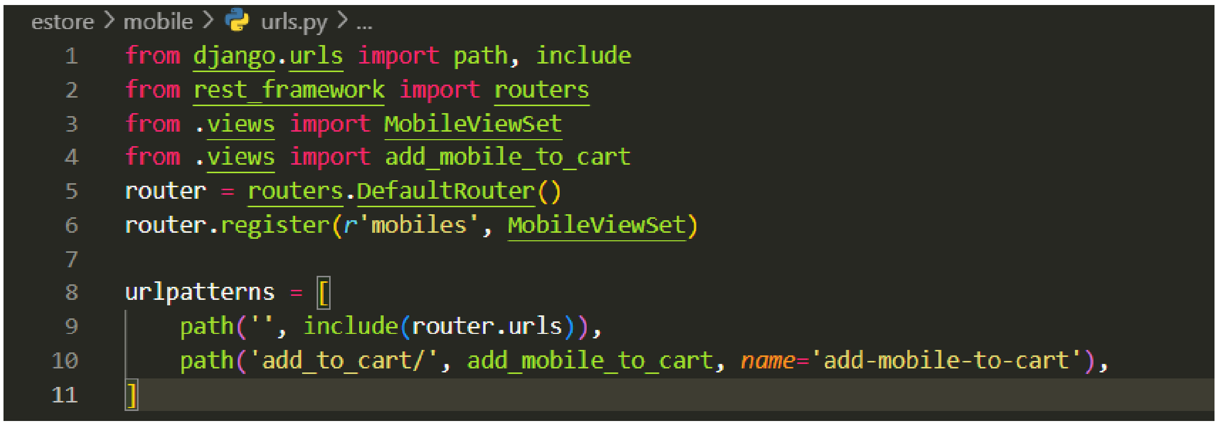
estore > mobile > routers.py > MongoDBRouter

```python
class MongoDBRouter:
    def db_for_read(self, model, **hints):
        if model._meta.app_label == 'mobile':
            return 'mongodb'
        return None

    def db_for_write(self, model, **hints):
        if model._meta.app_label == 'mobile':
            return 'mongodb'
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if (
            obj1._meta.app_label == 'mobile' or
            obj2._meta.app_label == 'mobile'
        ):
            return True
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == 'mobile':
            return db == 'mongodb'
        return None
```



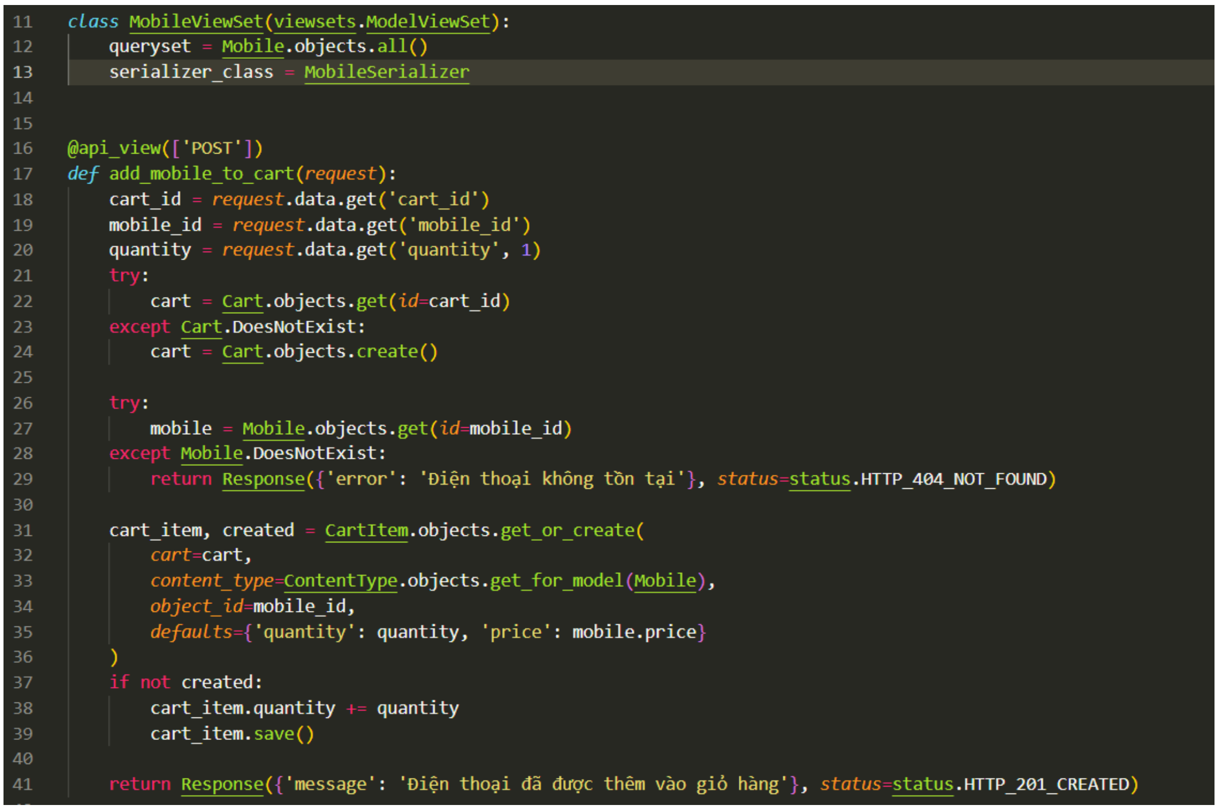
estore > mobile > serializers.py > ...

```python
from rest_framework import serializers
from .models import Mobile

class MobileSerializer(serializers.ModelSerializer):
    class Meta:
        model = Mobile
        fields = '__all__'
```



estore > mobile > urls.py > ...

```python
from django.urls import path, include
from rest_framework import routers
from .views import MobileViewSet
from .views import add_mobile_to_cart
router = routers.DefaultRouter()
router.register(r'mobiles', MobileViewSet)

urlpatterns = [
    path('', include(router.urls)),
    path('add_to_cart/', add_mobile_to_cart, name='add-mobile-to-cart'),
]
```



```python
class MobileViewSet(viewsets.ModelViewSet):
    queryset = Mobile.objects.all()
    serializer_class = MobileSerializer


@api_view(['POST'])
def add_mobile_to_cart(request):
    cart_id = request.data.get('cart_id')
    mobile_id = request.data.get('mobile_id')
    quantity = request.data.get('quantity', 1)
    try:
        cart = Cart.objects.get(id=cart_id)
    except Cart.DoesNotExist:
        cart = Cart.objects.create()

    try:
        mobile = Mobile.objects.get(id=mobile_id)
    except Mobile.DoesNotExist:
        return Response({'error': 'Điện thoại không tồn tại'}, status=status.HTTP_404_NOT_FOUND)

    cart_item, created = CartItem.objects.get_or_create(
        cart=cart,
        content_type=ContentType.objects.get_for_model(Mobile),
        object_id=mobile_id,
        defaults={'quantity': quantity, 'price': mobile.price}
    )
    if not created:
        cart_item.quantity += quantity
        cart_item.save()

    return Response({'message': 'Điện thoại đã được thêm vào giỏ hàng'}, status=status.HTTP_201_CREATED)
```